Dynamical Systems CW3

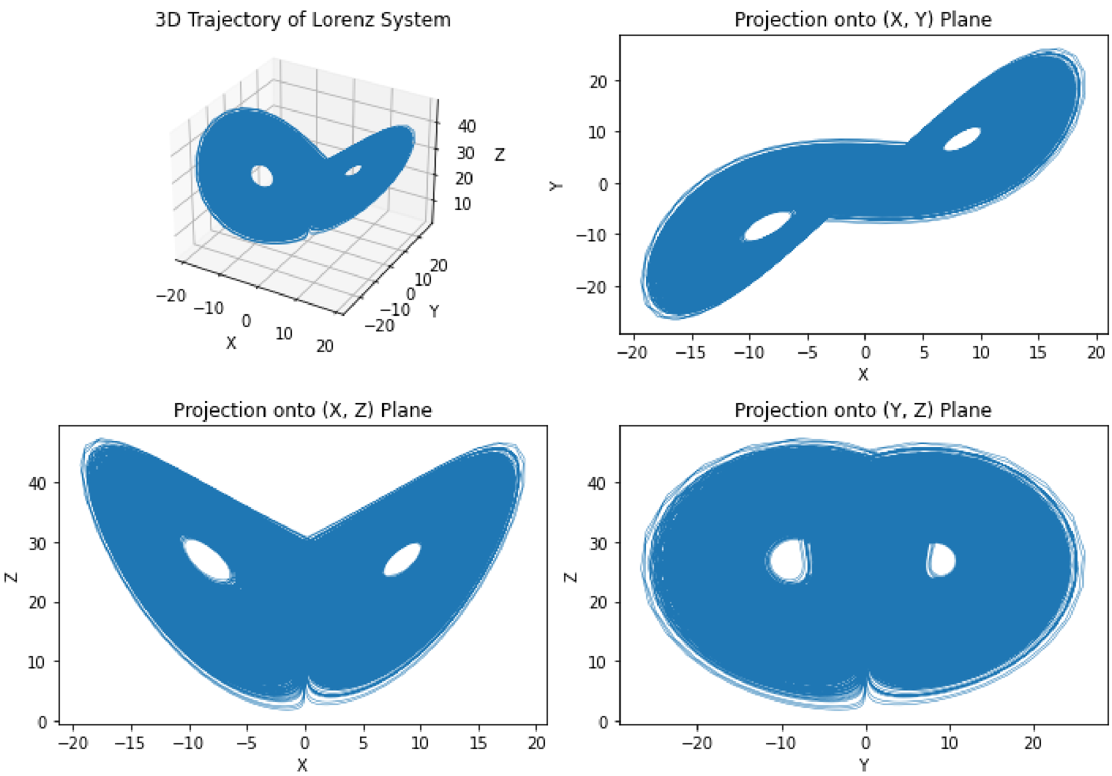
CID: 02044064

February 13, 2025

Q1a

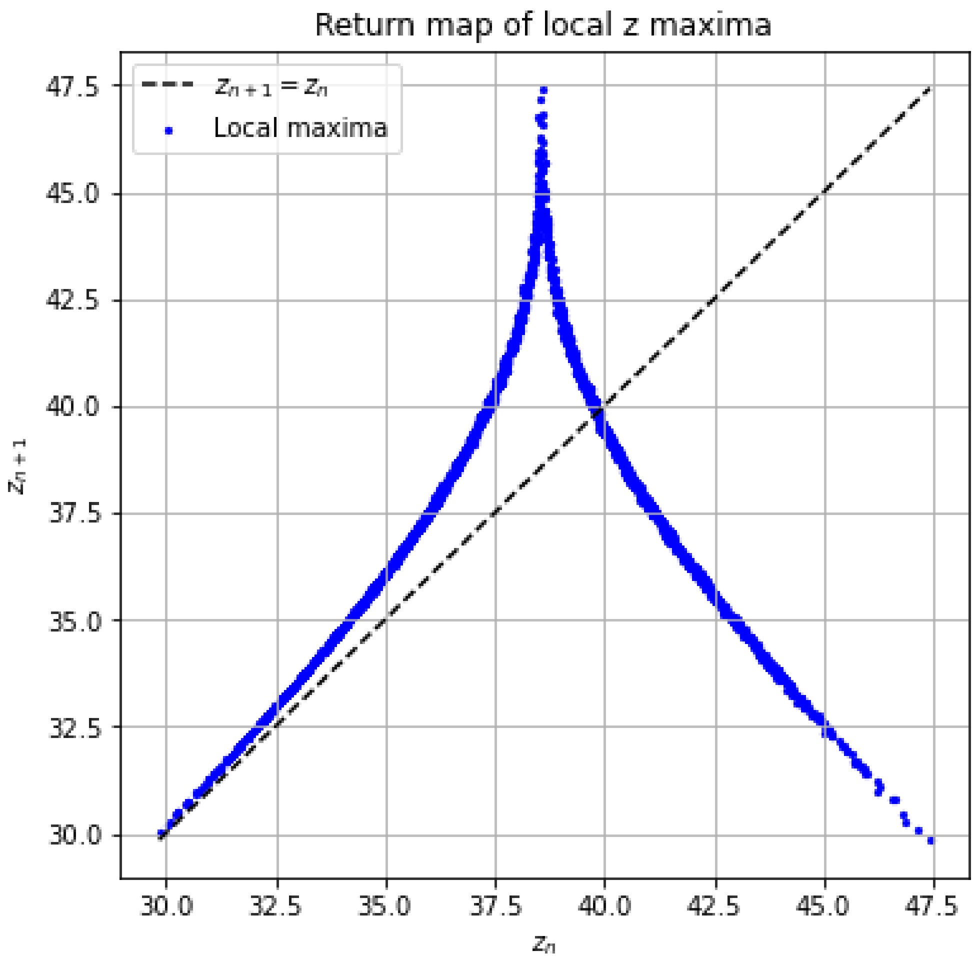
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.signal import find_peaks

# Define the Lorenz system
def lorenz(t, state, sigma=10, rho=28, beta=8/3):
    x, y, z = state
    dxdt = sigma * (y - x)
    dydt = x * (rho - z) - y
    dzdt = x * y - beta * z
    return [dxdt, dydt, dzdt]

# Time span 2500s
t_span = (0, 2500)
t_eval = np.linspace(t_span[0], t_span[1], 150000)

# Initial condition
initial_state = [1.0, 1.0, 1.0]

# Solve ODE
sol = solve_ivp(lorenz, t_span, initial_state, t_eval=t_eval, method='RK45')

# Extract solution
t, x, y, z = sol.t, sol.y[0], sol.y[1], sol.y[2]

# Transient removal (only keep data where t > 500s)
valid_indices = t > 500
t, x, y, z = t[valid_indices], x[valid_indices], y[valid_indices], z[valid_indices]

# Find local maxima of z(t)
peaks, _ = find_peaks(z)
z_maxima = z[peaks]

# Create figure and subplots
fig = plt.figure(figsize=(10, 7))

# Trajectory plot (Top-left)
ax1 = fig.add_subplot(221, projection='3d')
ax1.plot(x, y, z, lw=0.5)
ax1.set_xlabel("X")
```

```python
ax1.set_ylabel("Y")
ax1.set_zlabel("Z")
ax1.set_title("3D Trajectory of Lorenz System")

# Projection: (x, y) (Top-right)
ax2 = fig.add_subplot(222)
ax2.plot(x, y, lw=0.5)
ax2.set_xlabel("X")
ax2.set_ylabel("Y")
ax2.set_title("Projection onto (X, Y) Plane")

# Projection: (x, z) (Bottom-left)
ax3 = fig.add_subplot(223)
ax3.plot(x, z, lw=0.5)
ax3.set_xlabel("X")
ax3.set_ylabel("Z")
ax3.set_title("Projection onto (X, Z) Plane")

# Projection: (y, z) (Bottom-right)
ax4 = fig.add_subplot(224)
ax4.plot(y, z, lw=0.5)
ax4.set_xlabel("Y")
ax4.set_ylabel("Z")
ax4.set_title("Projection onto (Y, Z) Plane")

plt.tight_layout()
plt.show()
```



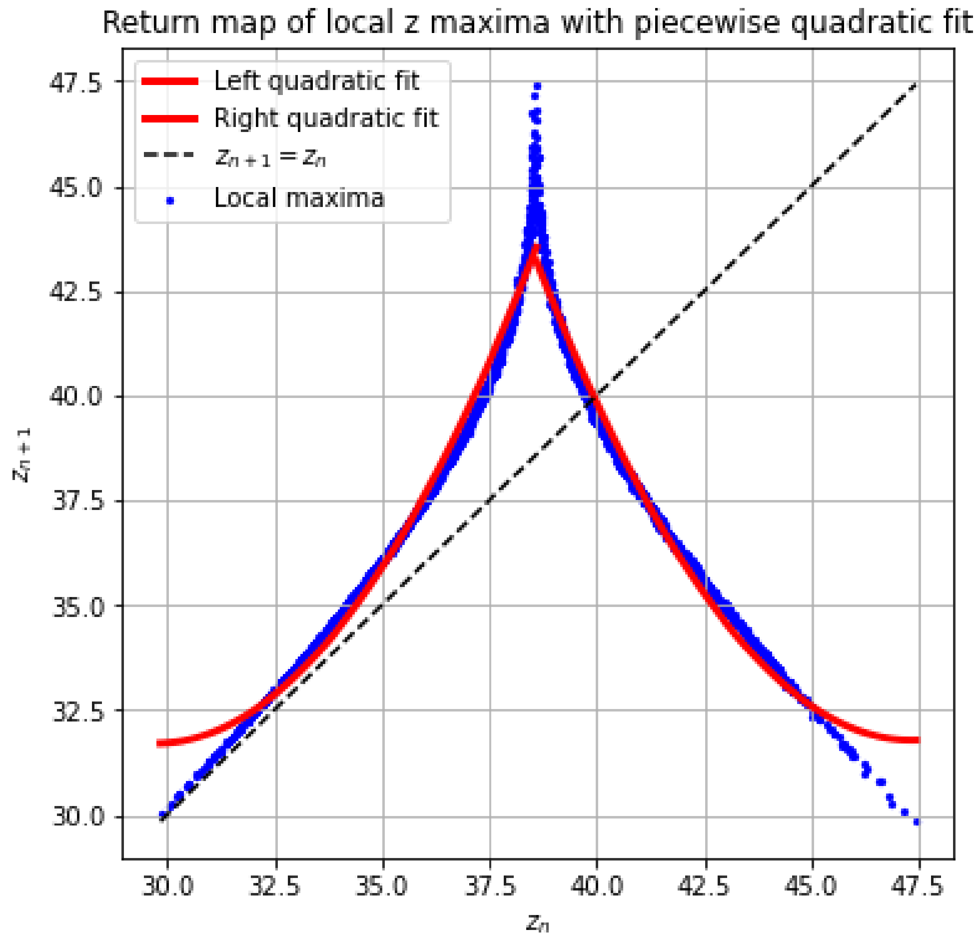
Q1b-c

```
# Return map
z_maxima_indices, _ = find_peaks(z)  # Find local maxima of z
z_maxima = z[z_maxima_indices]  # Extract z values at local maxima

plt.figure(figsize=(6, 6))
plt.scatter(z_maxima[:-1], z_maxima[1:], color='b', s=5, label="Local maxima")

# Add z_{n+1} = z_n dashed line
min_z = min(z_maxima)
max_z = max(z_maxima)
plt.plot([min_z, max_z], [min_z, max_z], 'k--', label=r"$z_{n+1}=z_n$")

plt.xlabel(r"$z_n$"), plt.ylabel(r"$z_{n+1}$")
plt.title("Return map of local z maxima")
plt.legend()
plt.grid()
plt.show()
```



Q1d

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks

# Compute local maxima of z
z_maxima_indices, _ = find_peaks(z)
z_maxima = z[z_maxima_indices]

# Identify peak of the return map (largest zn+1)
peak_index = np.argmax(z_maxima) - 1  # Shift back to go from z_n+1 to z_n
peak_value = z_maxima[peak_index]
peak_height = z_maxima[peak_index + 1]

# Create arrays of (zn, zn+1) pairs
zn = z_maxima[:-1]  # z_n values
zn1 = z_maxima[1:]  # z_{n+1} values

# Split into left and right based on peak_value
zn_left, zn1_left = zn[zn <= peak_value], zn1[zn <= peak_value]
zn_right, zn1_right = zn[zn > peak_value], zn1[zn > peak_value]

# Fit separate quadratic functions to both branches
coeffs_left = np.polyfit(zn_left, zn1_left, 2)  # Quadratic fit (left)
coeffs_right = np.polyfit(zn_right, zn1_right, 2)  # Quadratic fit (right)

# Generate smooth values for plotting quadratic fits
zn_left_smooth = np.linspace(min(zn_left), max(zn_left), 100)
zn_right_smooth = np.linspace(min(zn_right), max(zn_right), 100)

zn1_left_fit = np.polyval(coeffs_left, zn_left_smooth)
zn1_right_fit = np.polyval(coeffs_right, zn_right_smooth)

# Plot return map with piecewise quadratic fits
plt.figure(figsize=(6, 6))
plt.scatter(zn, zn1, color='b', s=5, label="Local maxima")

# Plot smooth quadratic fits
plt.plot(zn_left_smooth, zn1_left_fit, color='red', linewidth=3, label="Left quadratic fit")
plt.plot(zn_right_smooth, zn1_right_fit, color='red', linewidth=3, label="Right quadratic fit")

# Add z_{n+1} = z_n dashed line
min_z, max_z = min(z_maxima), max(z_maxima)
plt.plot([min_z, max_z], [min_z, max_z], 'k--', label=r"$z_{n+1}=z_n$")

plt.xlabel(r"$z_n$")
plt.ylabel(r"$z_{n+1}$")
plt.title("Return map of local z maxima with piecewise quadratic fit")
plt.legend()
plt.grid()
plt.show()
```



Q1e

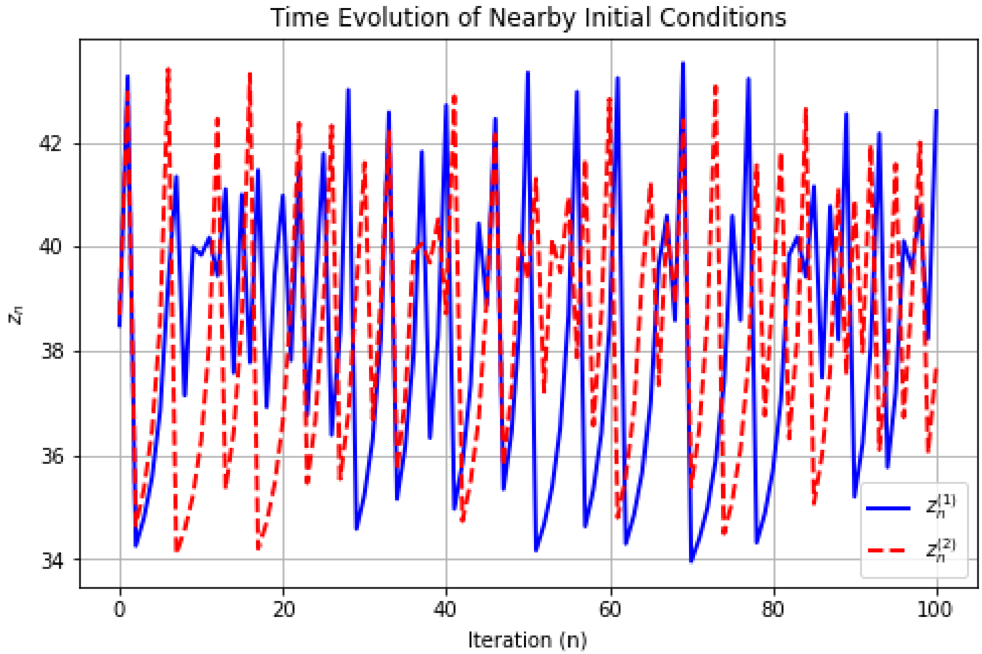
```python
# Extract coefficients
a_l2, a_l1, a_l0 = coeffs_left  # Left branch coefficients
a_r2, a_r1, a_r0 = coeffs_right  # Right branch coefficients

# Print the piecewise quadratic function
print(f"z_(n+1) = ")
print(f"({a_l2:.5f}) z_n^2 + ({a_l1:.5f}) z_n + ({a_l0:.5f}), if z_n  {peak_value:.5f}")
print(f"({a_r2:.5f}) z_n^2 + ({a_r1:.5f}) z_n + ({a_r0:.5f}), if z_n > {peak_value:.5f}")
```

```
z_(n+1) =
(0.15013) z_n^2 + (-8.92044) z_n + (164.20245), if z_n ≤ 38.58928
(0.14953) z_n^2 + (-14.15647) z_n + (366.82982), if z_n > 38.58928
```

Q1f

We will consider the time series plot for z and aim to identify (the lack of) periodic behaviour.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks

def piecewise_quadratic(z, z_peak, left_coeffs, right_coeffs):
    """ Computes the next iterate z_{n+1} based on the fitted piecewise quadratic function. """
    if z <= z_peak:
        return coeffs_left[0] * z**2 + coeffs_left[1] * z + coeffs_left[2]
    else:
        return coeffs_right[0] * z**2 + coeffs_right[1] * z + coeffs_right[2]

z1 = [peak_value - 0.1]  # Slightly left of the peak value
z2 = [peak_value + 0.1]  # Slightly right of the peak value

n_iter = 100
for _ in range(n_iter):
    z1.append(piecewise_quadratic(z1[-1], peak_value, coeffs_left, coeffs_right))
    z2.append(piecewise_quadratic(z2[-1], peak_value, coeffs_left, coeffs_right))

# Plot the results
plt.figure(figsize=(8, 5))
plt.plot(range(n_iter+1), z1, label=r"$z_n^{(1)}$", color='b', linewidth=2)
plt.plot(range(n_iter+1), z2, label=r"$z_n^{(2)}$", color='r', linewidth=2, linestyle='dashed')
plt.xlabel("Iteration (n)")
plt.ylabel(r"$z_n$")
plt.title("Time Evolution of Nearby Initial Conditions")
plt.legend()
plt.grid()
plt.show()
```



Since there is no periodic behaviour within the first 100 time steps, the behaviour is likely to be chaotic for the initial condition $z_0 = 32$ (to show that it is indeed chaotic, we could compute the Lyapunov exponents and check that the largest one is positive). Alternatively, we can characterise chaos by sensitivity to initial conditions and topological transitivity. Whilst topological transitivity is difficult to show numerically, we can see from the above plot that the system is sensitive to initial conditions as the z coordinates of the two trajectories start very close to each other but differ significantly over time.